Tampereen yliopisto

Arttu Karpale

BINÄÄRIDATAN VISUALISOINTI

Informaatioteknologia ja viestintä
Kandidaatintyö
Toukokuu 2019

TIIVISTELMÄ

Arttu Karpale: Binääridatan visualisointi
Kandidaatintyö
Tampereen yliopisto
Informaatioteknologia ja viestintä
Toukokuu 2019

Tässä opinnäytetyössä tarkastellaan visualisoinnin käyttöä takaisinmallinnuksen työkaluna. Takaisinmallinnuksessa käsitellään usein suuria määriä tietoa, jota on vaikea ymmärtää ilman abstraktiota. Työssä todetaan, että eri visualisointien avulla voidaan tunnistaa tietorakenteita tuntemattomasta datasta tehokkaasti. Jotkin visualisoinnin keinot todetaan toimivan eri tunnistamisen osa-alueissa paremmin kuin toiset.

Avainsanat: takaisinmallinnus, visualisointi, tietorakenteet

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

RGB	Eng. Red Green Blue
Takaisinmallintaminen	Eng. Reverse engineering
Tiedostomuoto	Eng. File format

1. JOHDANTO

Informaatiota on varastoitu digitaalisesti jo vuosikymmeniä ja suurta osaa maailman tiedosta ylläpidetään digitaalisesti. Teknologian kehittyessä tietoa voidaan varastoida entistä enemmän ja sen käsittely ihmisluettavassa muodossa vaikeutuu. Massatietoa tarkastellaan usein ohjelmistoilla, jotka käsittelevät tiedon helpommin ymmärrettävään muotoon ennen kuin se esitetään käyttäjälle. Nämä ohjelmistot perustuvat tiedostomuotoihin ja niiden asettamiin rakenteisiin. Tiedostomuotoja tulee käyttöön enemmän vuosittain ja nämä tiedostomuodot tarvitsevat uutta ohjelmistoa käsittelemään niiden sisäisiä tietorakenteita. Uudet tiedostomuodot ovat useasti monipuolisempia kuin aikaisemmat tiedostomuodot, mikä johtaa vanhojen tiedostomuotojen käyttämättömyyteen. Useita tiedostomuotoja ei kyetä enää käsittelemään, sillä niitä koskeva dokumentaatio on hukkunut syystä tai toisesta [1]. Puutteellinen dokumentaatio hankaloittaa tiedonkäsittelyohjelmiston tekoa ja voi aiheuttaa tiedon katoamisen. Tällaisessa tilanteessa voidaan yrittää takaisinmallinnusta ja selvittää mahdollisia käsittelytapoja, joilla saataisiin esille tiedon merkitys. Takaisinmallinnus on kuitenkin usein työlästä, varsinkin jos käsiteltävä tietorakenne on monimutkainen. [1] Takaisinmallinnusta voi helpottaa hyödyntämällä ihmisten kykyä hahmottaa muotoja ja yhtäläisyyksiä visualisoimalla tarkasteltavaa tietoa ja sen rakennetta [4, 5, 6].

Tässä työssä perehdytään tekniikkoihin, joilla voidaan käsitellä takaisinmallinnettavaa tietoa visualisoinnin avulla ihmisluettavassa muodossa. Työssä tutkitaan suurten tiedostojen käsittelyä visuaalisin menetelmin ja käydään läpi menetelmiin liittyviä hyötyjä.

Työn toisessa luvussa käsitellään takaisinmallinnusta ja sen käsitteitä tämän työn osalta. Kolmannessa luvussa alustetaan visualisoinnissa käytettyjä menetelmiä. Neljännessä luvussa esitetään eri visualisointialgoritmeja, joita voidaan käyttää analyysissä. Viidennessä luvussa paneudutaan algoritmien toimintaan ja niitä hyödynnetään esimerkkitapauksessa. Kuudennessa luvussa työn tuloksista luodaan yhteenveto.

2. TAKAISINMALLINNUS

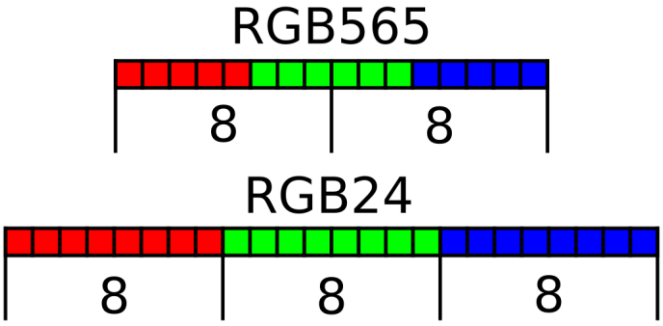
Keinotekoisten kappaleiden tutkimista ja niiden sisältävän informaation selvittämistä kutsutaan takaisinmallinnukseksi (reverse engineering). Kappaleeseen liittyvä informaatio voi koskea mitä tahansa osaa keinotekoisesta kokonaisuudesta. Takaisinmallinnuksen käyttökohteita ovat usein laitteet tai ohjelmistot joiden sisäisiä toimintoja halutaan tutkia. Esimerkiksi ohjelmisto voi olla takaisinmallinettavana, jos siihen viittaava dokumentaatio on hukkunut ja dokumentaatio halutaan entisöidä. Myös esimerkiksi tuotteiden tietoturvallisuutta voidaan tutkia takaisinmallintamalla mahdollisia haavoittuvuuksia. [1]

Ohjelmistot sisältävät usein suuria määriä dataa ja data kuvastaa niin lyhyitä arvoja kuin monimutkaisia tietorakenteita. Tunnetuille ohjelmistoarkkitehtuureille ja tietorakenteille on olemassa takaisinmallinnustyökaluja, jolloin tiedon suuri määrä ei koidu haitaksi, vaan halutut ominaisuudet voidaan löytää automaattisesti. Työkalut auttavat tarkasteltavan tiedoston sisäisten tietorakenteiden löytämisessä ja käsittelyssä. Automaattisten työkalujen kehitystä varten tarvitaan kuitenkin alustavaa dokumentaatiota ohjelmistosta, jotta ohjelmiston eri tietorakenteita voidaan löytää sekä lukea tiedostosta. Jos dokumentaatio on hukkunut tai se on vaikeasti saatavilla, voidaan takaisinmallinnuksella myös tutkia ja yrittää entisöidä ohjelmistoa koskevaa dokumentaatiota. Tässä tapauksessa takaisinmallinnus tehdään matalalla tasolla ja vaatii takaisinmallintajalta kokemusta monista eri käytännöistä sekä ohjelmistotuotantotavoista. [1]

Matalan tason takaisinmallinnuksessa vaikeudeksi voi koitua eri tietorakenteiden havaitseminen toisten tietorakenteiden joukosta. Havaitsemista voidaan suorittaa automaattisesti, mutta työkalujen tarkkuus riippuu aikaisemman dokumentaation tarkkuudesta. On mahdollista, ettei automaattiset havainnointitavat toimi halutulla tarkkuudella, jos takaisinmallinnuksen kohde on huonosti tunnettu. [2] Tietorakenteiden havaitsemista voidaan myös suorittaa työkaluilla, jotka eivät vaadi aikaisempaa informaatiota [1]. Tällöin takaisinmallinnuksessa yritetään koostaa tietoa helpommin käsiteltävään muotoon, esimerkiksi entropian avulla [2, 3]. Kaksi hyvin samankaltaista tietorakennetta on kuitenkin lähes mahdoton erottaa toisistaan pelkän entropian avulla, joten työkaluja on usein oltava monia [1, 2]. Dataa voidaan myös muuttaa ihmisluettavaan muotoon, muuttamalla yksittäisiä tietoalkioita visuaalisiksi elementeiksi [4, 5, 6]. Saadun visualisoinnin avulla, takaisinmallintaja voi tehdä johtopäätöksiä tiedosta ja sen rakenteesta helpommin, kuin että hän olisi tutkinut dataa vain matalan tason työkaluilla [4, 5, 6].

3. TIETOALKIOT VISUALISOINNISSA

Digitaalisessa mediassa tallennukseen käytetään bittejä, ja ne voidaan jakaa tietyn pituisiksi tietoalkioiksi kuvaamaan jotakin suurempaa arvoa. Esimerkiksi RGB565-formaattia käyttävässä kuvassa olevan kuvapisteen väriarvoja osoitetaan 16-bittisellä arvolla, joka on jaettu kolmeen eri kokoiseen tietoalkioon. RGB565-formaatissa alkiot ovat 5 bitin pituisia punaiselle ja siniselle värille tai 6 bittiä pitkiä vihreälle. Toisaalta samaa kuvapistettä voitaisiin esittää RGB24-formaatilla, jossa kuvapisteen väriarvoja kuvataan 24-bittisellä arvolla. RGB24-formaatissa jako tapahtuu tasaisesti 8 bitin alkioihin jokaista väriä kohden. [7] Tietoalkioita voidaan formaatista riippumatta tarkastella jollakin tasaisella tarkasteluvälillä. Tämän tarkasteluvälin sisällä olevista biteistä voidaan muodostaa tarkastelualkio. Kuvasta 1 voidaan havaita formaattien erot, kun niitä tarkastellaan 8 bittisten tarkastelualkioiden kautta. RGB565-formaatissa kahteen tarkastelualkioon vaikuttaa vihreä väri, kun taas RGB24-formaatissa jokainen tarkastelualkio kuvaa omaa väriään.



Kuva 1. *RGB-formaattien vertailu*

Tarkastelualkion kautta tiedon rakenteeseen ei tarvitse suoraan ottaa kantaa, vaan tarkastelu voidaan suorittaa formaatista riippumatta, vaikka datassa esiintyisi esimerkiksi normaalia lukua haittaavia virheitä. Tarkastelualkioiden avulla voidaan myös verrata eri tietoalueita toisiinsa ja päätellä ovatko ne samaa tietotyyppiä. Visualisointia hyödyntämällä voidaan informaatiota sisältävästä datasta saada esiin mahdollista rakenteellista tietoa. [2, 4]

Kahta samaa rakennetta noudattavaa tietorakennetta voidaan tarkastelualkion avulla vertailla, vaikka tietorakenteen sisäiset alkiot eivät olisikaan yhtä suuria tarkastelualkion kanssa [2]. Esimerkiksi kaksi RBG565-rakennetta jaettuna kahteen 8 bittiseen tarkastelualkioon käyttäytyvät samalla tavalla, kun niiden arvoja muutetaan. Esimerkiksi jos vihreää väriä muutetaan kuvassa jollakin tavalla, muuttuvat myös tarkastelualkioiden arvot säännöllisesti ja pysyvät ne vertailukelpoisina toisiinsa nähden, vaikka tarkastelualkioiden arvot eivät kuvaakaan tietorakenteen oikeellista arvoa.

Tarkastelualkioita voidaan muodostaa monin eri menetelmin. Tarkastelualkion arvot voivat kuvastaa tietovirran arvoja tai ne voivat esimerkiksi kuvastaa tietyn alueen entropiaa tietovirrassa. Informaatiota tallentaessa tai kommunikoidessa on noudatettava jotakin järjestystä tai rakennetta, sillä muuten informaatio voitaisiin sekoittaa ulkoiseen kohinaan. Alkioiden satunnaisuudesta voidaan tehdä arvo, joka kuvastaa eri alkioiden todennäköisyyttä esiintyä annetulla alueella. [3] Todennäköisyyden avulla voidaan muodostaa entropia-arvo annetulle alueelle, joka kuvastaa alueen satunnaisuutta. Satunnaiselta vaikuttavasta datasta voidaan entropiaa hyödyntäen tutkia, että onko se oikeasti täysin satunnaista vai noudattaako se mahdollisesti jotakin tietorakennetta [2]. Monesti kuitenkin tiedostoissa ja tietokannoissa tieto ei ole yksikäsitteistä, vaan dataa voidaan käsitellä monessa kontekstissa eri tavoin [1]. Jotkin tietorakenteet pitävät sisällään monia pienempiä tietorakenteita ja niiden erittely voi olla vaikeaa, jos kaksi samankaltaista tietoaluetta varastoidaan vierekkäin. Tietorakenteita voidaan kuitenkin jakaa karkeasti niiden entropian mukaan [2, 6].

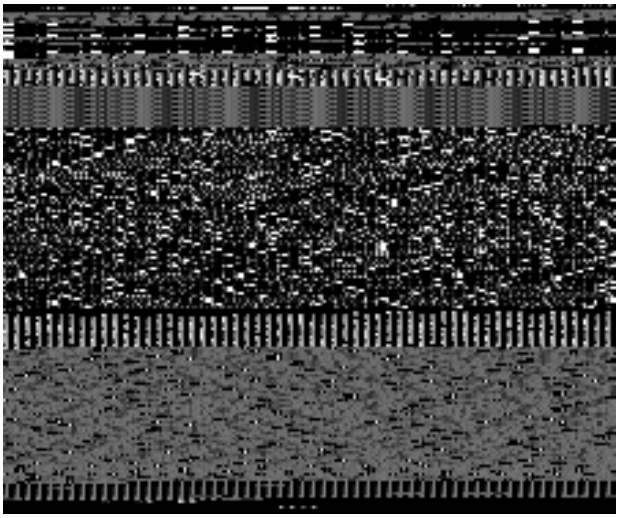
4. VISUALISOINTIALGORITMIT

Takaisinmallinnuksessa läpikäytävän datan määrä voi koitua valtavaksi ja ihmisluettavuuden kannalta epäkäytännölliseksi [2]. Tietoa voidaan pakata kuviin visualisointialgoritmeja hyödyntäen, jotta tietoa voitaisiin käsitellä ihmisluettavammassa muodossa suurempia määriä kerralla. Visualisointialgoritmissa tietoalkioita käsitellään niin, että niistä voidaan muodostaa visuaalista informaatiota. [2, 4]

Tarkastelualkiot voidaan sijoittaa eri ulotteisiin ympäristöihin tai niiden arvoista voidaan tehdä koosteita [4 ,6]. Sijoituksen tapauksessa jokainen tarkastelualkio kuvastaa yksittäistä kuvapistettä [2]. Koosteissa yksi kuvapiste voi koostua monen tarkastelualkion arvoista ja voi kuvastaa esimerkiksi arvojakaumaa [6].

4.1 Lineaarinen asettelu

Yksinkertainen tapa asetella tietoalkioita on järjestää ne riveittäin ja kasata rivit päällekkäin yhtenäiseksi kuvaksi. Jokainen tietoalkio muutetaan kuvapisteeksi, joka on sitä kirkkaampi mitä lähempänä tietoalkion arvo on sen maksimiarvoa. [2] Esimerkiksi 8 bittinen tietoalkio näkyy täysin valkoisena, jos sen arvo on 255, ja täysin mustana, jos sen arvo on 0.



Kuva 2. *Lineaarinen asettelu eri tietorakenteista*

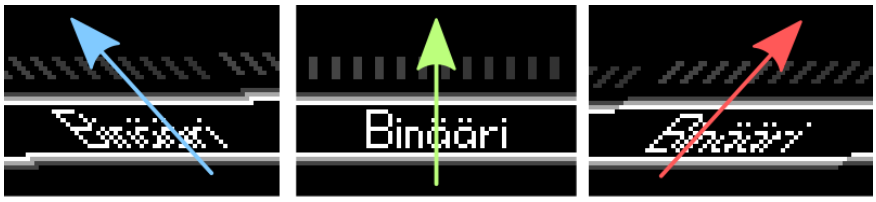
Lineaarinen asettelu mahdollistaa eri tietoalueiden tarkastelun ja sillä voidaan erotella tietorakenteita toisistaan [2]. Esimerkiksi kuvasta 2 voidaan nähdä, kuinka tiedoston eri tietorakenteet eroavat toisistaan. Jotkin rakenteet ovat jaksollisia ja ne näkyvät kuvassa

toistuvina pystyviivoina. Rakenteet myös käyttävät eri arvoalueita, jolloin niiden väritys-kin on eroavaa toisista tietorakenteista.

Matemaattisesti kuvapisteen koordinaatit voidaan laskea seuraavasti:

$$\mathrm{x} = mod_w i, \quad y = \lfloor i/w \rfloor,$$

jossa i on alkion paikka yksiulotteisessa tietovirrassa ja w on asettelun haluttu leveys. Leveyttä muuttamalla voidaan päätellä rakenteiden pituuksia. Toistuva tietorakenne näkyy yhtenäisenä rakenteena kuvassa, kun asettelun leveys on yhtä suuri tietorakenteen pituuden kanssa. Visualisointi vääristyy, jos leveys on pienempi tai suurempi kuin tietorakenteen pituus. Tämä vääristymä johtuu siitä, että visualisoinnin jokaisella rivillä kuvapisteitä piirretään riville leveyden ja pituuden välisen erotuksen verran liikaa tai liian vähän. [4]

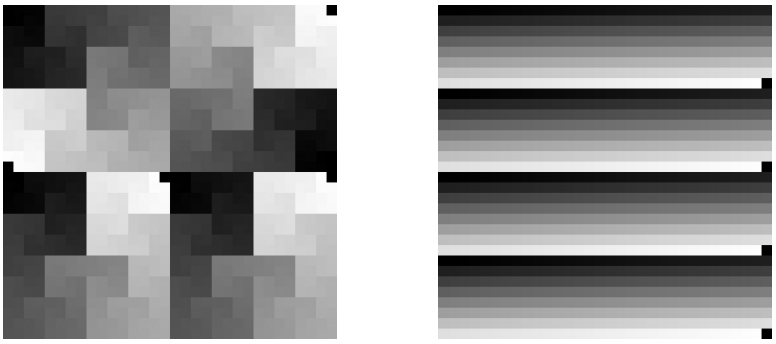


Kuva 3. *Leveyden vaikutus lineaarisessa asetelmassa*

Kuvassa 3 on demonstroitu vääristymää ja voidaan huomata, kuinka vääristymä esiintyy lineaarisessa asettelussa, kun asettelun leveys on joko liian pieni tai liian suuri. Erilaista vääristymää esiintyy myös, jos visualisoinnin leveys on jaollinen tietorakenteen pituudella. Tällöin yhteen lineaarisen asetelman riviin mahtuu tietorakenteen alkioita tasaisesti ja asetteluun muodostuu pystyviivoja. Tämä näkyy kuvassa 3 harmaina pystyviivoina. Jokainen tietorakenteen alkio vie enemmän kuin yhden rivin visualisoinnista, jos tietorakenteen pituus on jaollinen visualisoinnin leveydellä. Vääristymää on vaikeaa tutkia, jos tietorakenne ei ole vakiomittainen, kuten esimerkiksi merkkijonot tekstissä.

4.2 Hilbert-kurvi

Alkion etäisyys sen naapurialkioihin voi muuttua, kun tietoalkiota käytetään diskreetin kaksi ulotteisen koordinaatiston pisteenä. Lineaarisessa asettelussa leveyden ääripäissä etäisyys on huomattavasti suurempi alkion naapureihin, kuin asettelun muissa kohdissa. Naapurialkioiden etäisyyksien minimoimiseksi voidaan käyttää Hilbert-kurvia täyttämään kaksiulotteista koordinaatistoa kuvapisteillä. [8]

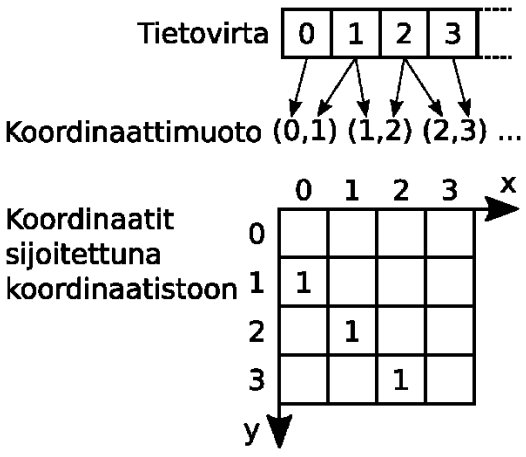


Kuva 4. *Vertailu Hilbert-kurvin ja lineaarisen asettelun välillä*

Kuvassa 4 on esitetty lineaarisesti kasvavaa dataa Hilbert-kurvin mukaisesti aseteltuna sekä lineaarisesti aseteltuna. Vertailusta voidaan huomata, kuinka lineaarisessa asettelussa esiintyy rivien välillä hyppyjä väriarvoissa. Hilbert-kurvissa kuitenkin kirkkaus muuttuu tasaisesti kurvin mukaisesti ja pysyy naapurialkioidensa lähellä. Hilbert-kurvia voidaan käyttää esimerkiksi entropian kanssa löytämään haittaohjelmien pakattuja osia ohjelmistojen sisältä [5].

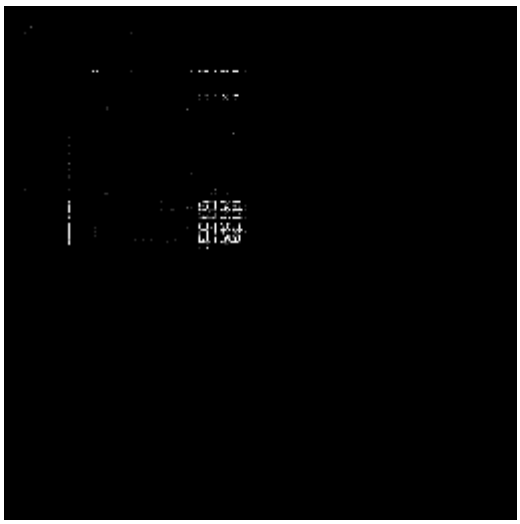
4.3 Digrafi

Tietoalkioita voidaan käsitellä n-ulotteisina koordinaatteina. Digrafin tapauksessa kahta alkiota käytetään muodostamaan kaksiulotteinen koordinaatti, joka voidaan piirtää kuvapisteenä. Jokainen kuvapiste digrafissa vastaa siirtymää arvosta A arvoon B, jos tietoalkiot otetaan järjestyksessä datasta niin, että jokainen alkio toimii x- ja y-koordinaatteina kerran. Kun alkioiden välinen etäisyys eli askelväli on 1, ovat ne toistensa naapureita ja kuvastavat arvojen välitöntä muutosta tiedossa. [6]



Kuva 5. *Digrafin muodostaminen*

Kuvassa 5 on esitetty digrafin muodostamisen vaiheet. Tietovirrasta otetaan kaksi alkiota kerrallaan tarkasteltavaksi. Ensimmäinen alkio toimii x-koordinaattina ja seuraava y-koordinaattina, ja näistä kahdesta alkiosta muodostetaan koordinaattipari. Tämän jälkeen tietovirrassa siirrytään yksi alkio eteenpäin ja muodostetaan seuraava koordinaattipari. Tällöin jokainen alkio toimii kerran niin x-koordinaattina, kuin y-koordinaattina ainakin kerran jossakin parissa. Ensimmäinen alkio kuitenkin toimii vain x-koordinaattina ja tietovirran viimeinen alkio toimii vain y-koordinaattina, sillä niillä ei ole edeltävää tai seuraavaa alkiota, jota käyttää koordinaattiparin muodostuksessa. Saaduista koordinaattipareista muodostetaan kuva, jossa jokaisen kuvapisteen kirkkaus kuvastaa jonkin koordinaattiparin lukumäärää kaikkien parien joukosta. Tässä työssä arvot alkavat 0:sta kuvan vasemmassa ylänurkassa ja vasenta ylänurkkaa kutsutaan origoksi. Origon lävistää vaakasuunnassa x-akseli ja pystysuunnassa y-akseli. Digrafin leveys sekä korkeus ovat riippuvaisia alkion koosta. [6]



Kuva 6. *Digrafi lyhyestä tekstistä*

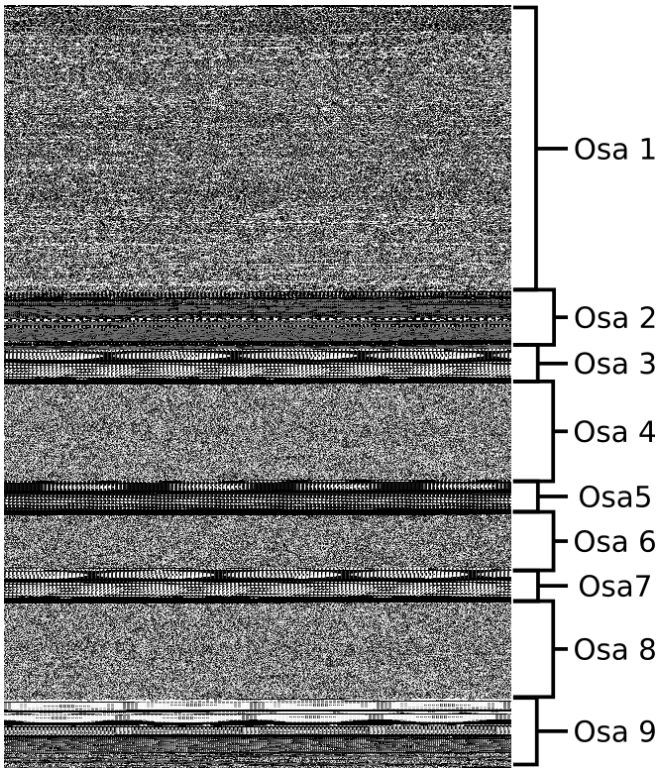
Digrafiin muodostuu tietorakenteille ominaisia piirteitä, joita voidaan verrata toisiin digrafeihin ja arvioida tietorakenteiden samankaltaisuuksia [6]. Esimerkiksi tekstistä tehdyssä digrafissa kirjoitettavien merkkien arvoalueelle muodostuu neliö, kuten kuvassa 6 on illustroitu. Latinalaista merkistöä käyttävä teksti näkyy neliönä, sillä tekstin esitysmuodossa käytetään yhtä 8 bitin pituista arvoa esittämään yhtä kirjoitettavaa merkkiä [9]. Tarkasteltavien tavujen arvot asettuvat usein 41 ja 7A heksadesimaaliarvojen välille, koska tekstissä käytetään enemmän isoja sekä pieniä kirjaimia kuin esimerkiksi erikoismerkkejä tai numeroita.

5. TUNTEMATTOMAN DATAN ANALYYSI

Käytetyimmistä tiedostomuodoista on saatavilla runsaasti mallidataa, jota voi käyttää otsakkeiden, tietorakenteiden sekä entropian tutkimiseen [1]. Täysin tuntemattoman datan analysoinnissa mallia ei ole saatavilla, minkä vuoksi yksittäisten tietorakenteiden tunnistaminen vaikeutuu. Visualisoinnin avulla yksittäiset tietorakenteet voidaan erotella toisistaan helpottaen takaisinmallinnusta [2].

Esimerkkitapauksessa tutkimme, kuinka eri visualisointialgoritmeja voidaan käyttää takaisinmallinnuksessa tunnistamaan tietorakenteita. Esimerkkitapauksen kohteeksi valittiin Mozilla Firefox -ohjelma ja siitä selvitetään missä eri tietorakenteet sijaitsevat sekä niiden mahdolliset merkitykset ohjelman toiminnassa [10]. Ohjelmiston versio on 60.6.1esr (32-bit). Visualisoinnit tehtiin C++-ohjelmointikielellä yksinkertaisiksi bittikartoiksi. Entropiaa käyttävät visualisoinnit tehtiin binvis.io-verkkosivun avulla [5].

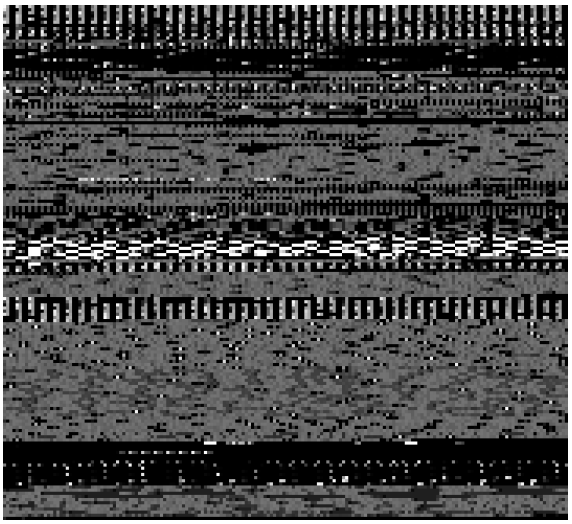
Ensiksi tarkasteltiin koko tiedoston rakennetta lineaarisella asettelulla. Tarkastelussa käytetyn tietoalkion pituudeksi valittiin 8 bittiä, sillä yhden tavun pituus on ISO/IEC-standardin mukaan yleensä 8 bittiä, jolloin standardia noudattavat tietorakenteet voidaan mahdollisesti huomata helpommin [11].



Kuva 7. *Firefox-ohjelman lineaarinen asettelu*

Kuvassa 6 on koko ohjelman kattava lineaarinen asettelu ja sen oikealle puolelle on kartoitettu suuret tietorakenteet asettelun avulla. Kuvasta 7 saadun osittelun mukaan tarkasteltiin yksittäisitä osista tehtyjä lineaarisia asetteluja tarkemmin. Osista 1, 4, 6 sekä 8 ei lineaarisen asettelun avulla löytynyt selviä rakenteita.

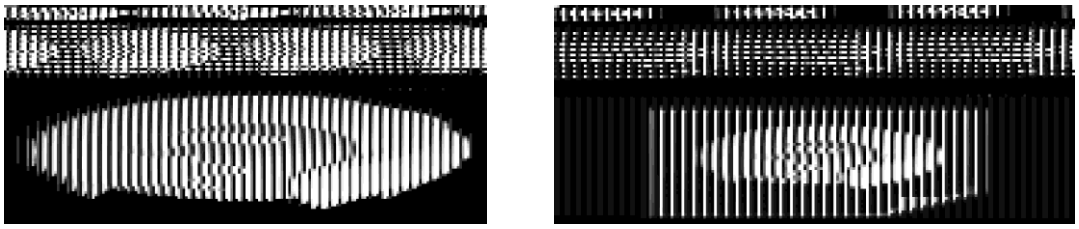
Osa 2 sisältää muutamaa eri tietorakennetta. Kuvasta 8 voidaan huomata, että suurin osa tiedosta on tasaisen harmaata. Tarkemmin tutkittuna harmaat alueet paljastuvat englanninkieliseksi tekstiksi.



Kuva 8. *Lineaarinen asettelu osasta 2*

Kuvasta voidaan myös huomata, että tekstin lähellä on toistuvia tietorakenteita. Nämä toistuvat tietorakenteet koostuvat 4 tavun pituisista alkioista ja alkiot muodostavat kuvaan pystyviivoja, sillä asettelun leveys on 4:llä jaollinen. Toistuvaa tietorakennetta esiintyy osan 2 alussa ja keskivaiheilla. Tämä rakenne sisältää luultavasti 32-bittisiä arvoja taulukkona ja voi mahdollisesti viitata jotenkin tietorakenteiden jälkeen tulevaan tekstiin. Viimeiseksi voidaan huomata osan keskellä oleva kirkas valkoinen tietorakenne. Se on 16 tavun välein toistuva tietorakenne, mutta sen alkioiden pituus ei ole vakio koko rakenteessa, vaan jotkin alkiot ovat vain 8 tavun pituisia. Tämäkin tietorakenne voi mahdollisesti säilyttää taulukkona jotakin dataa.

Osat 3, 5 ja 7 osoittautuvat samankaltaisiksi rakenteiksi, joista osat 3 ja 7 ovat lähes identtisiä. Rakenteet paljastuvat kuviksi lineaarisen asettelun avulla. Kuva on nähtävissä lineaarisesta asettelusta, kun asettelun leveys on kuvan levyinen. Jokaisesta osasta löytyy kolme kuvaa leveyksillä 64, 128 sekä 192.

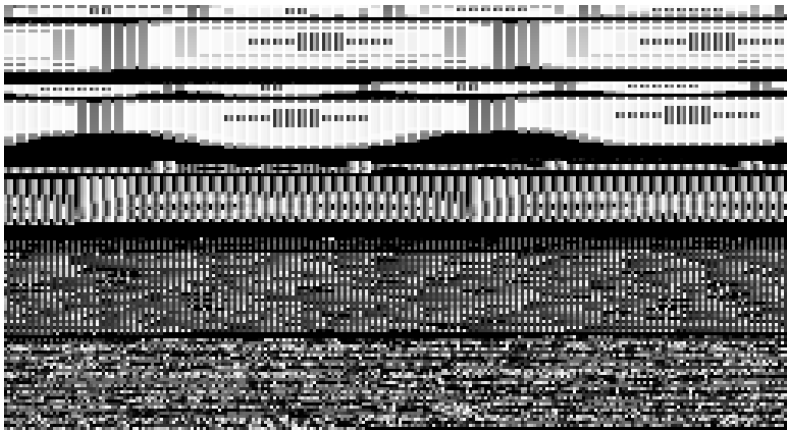


Kuva 9. *Osien 3 ja 5 lineaariset asetelmat. Asetelmien leveys on 192 tavua.*

Yhden osan sisältämät kolme kuvaa ovat eri kokoisia, mutta muuten samankaltaisia toisiinsa nähden. Kuvasta 9 nähdään, että osat 3 sekä 7 muistuttavat kuvassa 10 esitettyä Firefox-logoa [10]. Osa 5 kuvastaa html-tiedostoikonia, kun Firefox on ensisijainen selain Windows-käyttöjärjestelmässä. Kuvien yksittäiset pisteet ovat tallennettu 4 tavun pituisiin alkioihin, viitaten RGBA32-formaatin käyttöön [7].

Kuva 10. *Firefox-ohjelman logo*

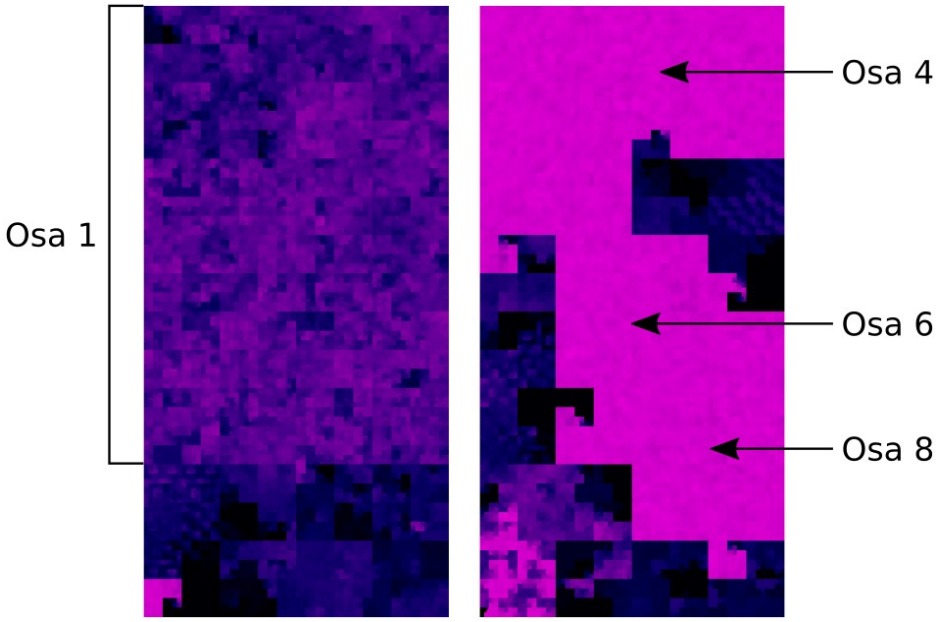
Osa 9 koostuu monesta tietorakenteesta. Kuvasta 11 nähdään, että ensimmäiseksi osassa on kuvaa muistuttava tietorakenne. Kuvarakenteita on kolme ja kahdessa esiintyy risti. Kolmas kuva muistuttaa silmänaamiota. Tässä tapauksessa yhdessä kuvarakenteessa on vain kaksi eri kokoa kuvalle, kun taas osissa 3, 5 ja 7 eri kokoja löytyi kolme.



Kuva 11. *Osan 9 lineaarinen asettelu*

Seuraava rakenne muodostuu 16-bittisistä arvoista. Arvon ensimmäinen tavu on lähes vakio, mutta kasvaa tietorakenteen loppua kohden. Toinen tavu kasvaa huomattavasti enemmän arvojen välillä lähes lineaarisesti. Tämä rakenne voi olla jokin taulukko esimerkiksi koodissa tehdyille hypyille tai taulukko joistakin ulkoasun kannalta tärkeistä arvoista. Viimeinen rakenne koostuu tekstiä muistuttavista pätkistä ja satunnaisista arvoista.

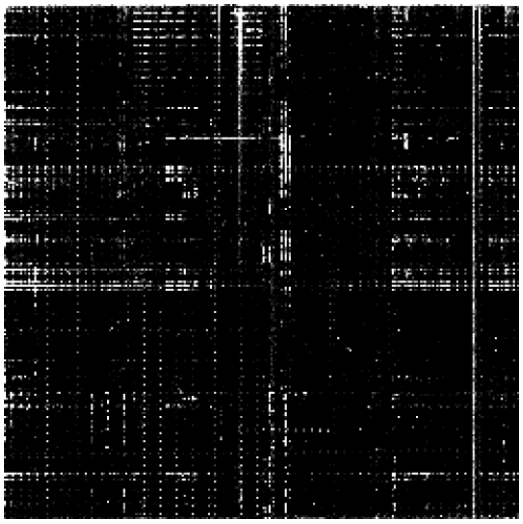
Lineaaristen asettelujen jälkeen tarkasteltiin koko ohjelmaa Hilbert-kurvin avulla käyttäen entropiaa tarkastelualkioiden arvoina. Entropia on suurimmillaan osissa 4, 6 sekä 8. Korkea entropia voi viitata kryptattuun tai pakattuun dataan [5]. Osa 1 poikkeaa huomattavasti osista 4, 6 ja 8, vaikka ne näyttävät lineaarisessa asettelussa samankaltaisilta.



Kuva 12. *Hilbert-kurvi tutkittavan tiedoston entropiasta*

Huomattavaa Hilbert-kurvin tuottamassa visualisoinnissa on se, että tietyn tietorakenteen paikantaminen tästä asetelmasta on huomattavasti vaikeampaa kuin lineaarisessa asettelussa. Kuvasta 12 voidaan huomata kuinka osat 4, 6 sekä 8 sulautuvat yhdeksi korkean entropian alueeksi Hilbert-kurvissa, vaikeuttaen tarkkaa erottelua näiden tietorakenteiden välillä. Eri osia tutkittiin myös yksittäin Hilbert-kurvia käyttäen, mutta sen avulla oli vaikeaa saada selville osista tarkkaa tietoa.

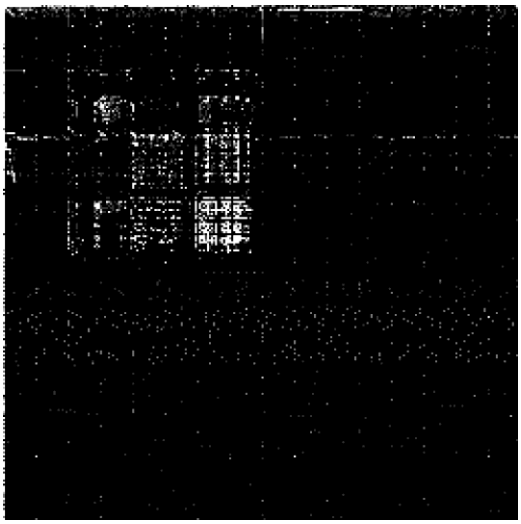
Hilbert-kurvin jälkeen osia tutkittiin digrafin avulla. Osa 1 näyttää lineaarisessa asettelussa lähes satunnaiselta datalta, mutta digrafin ja Hilbert-kurvin avulla voidaan todeta, että tieto ei ole satunnaista vaan noudattaa jotakin tietorakennetta. Kuvasta 13 voidaan nähdä kuinka arvot pakkautuvat tietyille arvoille muodostaen viivoja.



Kuva 13. *Digrafi osasta 1*

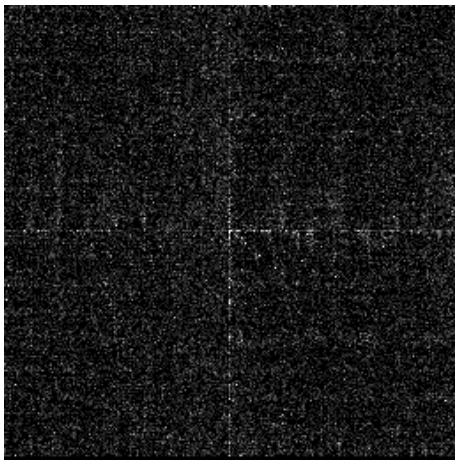
Osa 1 vaikuttaa ohjelmakoodilta, sillä sen käytetyt arvot jakautuvat toistuvasti tietyille arvoalueille. Osa näistä arvoalueista voi olla usein käytettyjä konekäskyjä ja viivat muodostuvat muuttujien vaikutuksesta. Varsinkin kuvassa 8 oikealla oleva täysin valkoinen pystyviiva voi olla useasti kutsuttava käsky. Arvoja tutkimalla voitaisiin tarkemmin päätellä, onko tieto oikeasti ohjelmakoodia ja mitä arkkitehtuuria varten se on tehty.

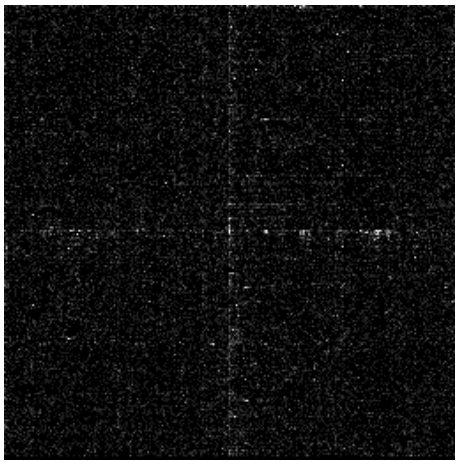
Osasta 2 löytyvä teksti on huomattavissa myös digrafissa. Kuvassa 14 on nähtävissä kuvassa 5 esiintyvä latinalaiselle merkistölle ominainen neliö, mutta osassa 2 käytetään huomattavasti enemmän erikoismerkkejä ja isoja kirjaimia, jolloin kuvaan muodostuu kolme neliötä ominaisen neliön lisäksi. Loput rakenteista muodostavat säännöllisiä kuvioita digrafiin, kuten digrafin alemmassa osassa esiintyvästä vyöhykkeestä voidaan huomata.

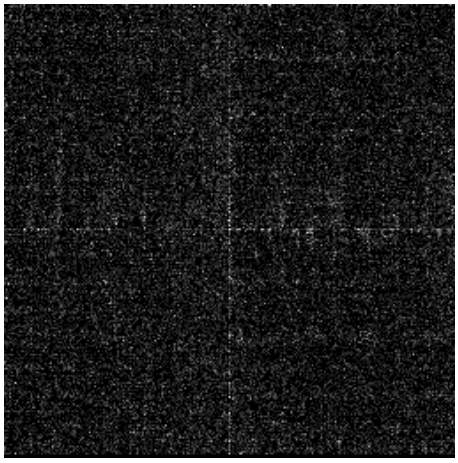


Kuva 14. *Digrafi osasta 2*

Osat 4,6 sekä 8 osoittautuvat samankaltaisiksi tietorakenteiksi. Digrafeista voidaan todeta, että osat noudattavat samankaltaista rakennetta, jossa alkiot saavat arvoja tavun koko arvoalueelta.

4) 

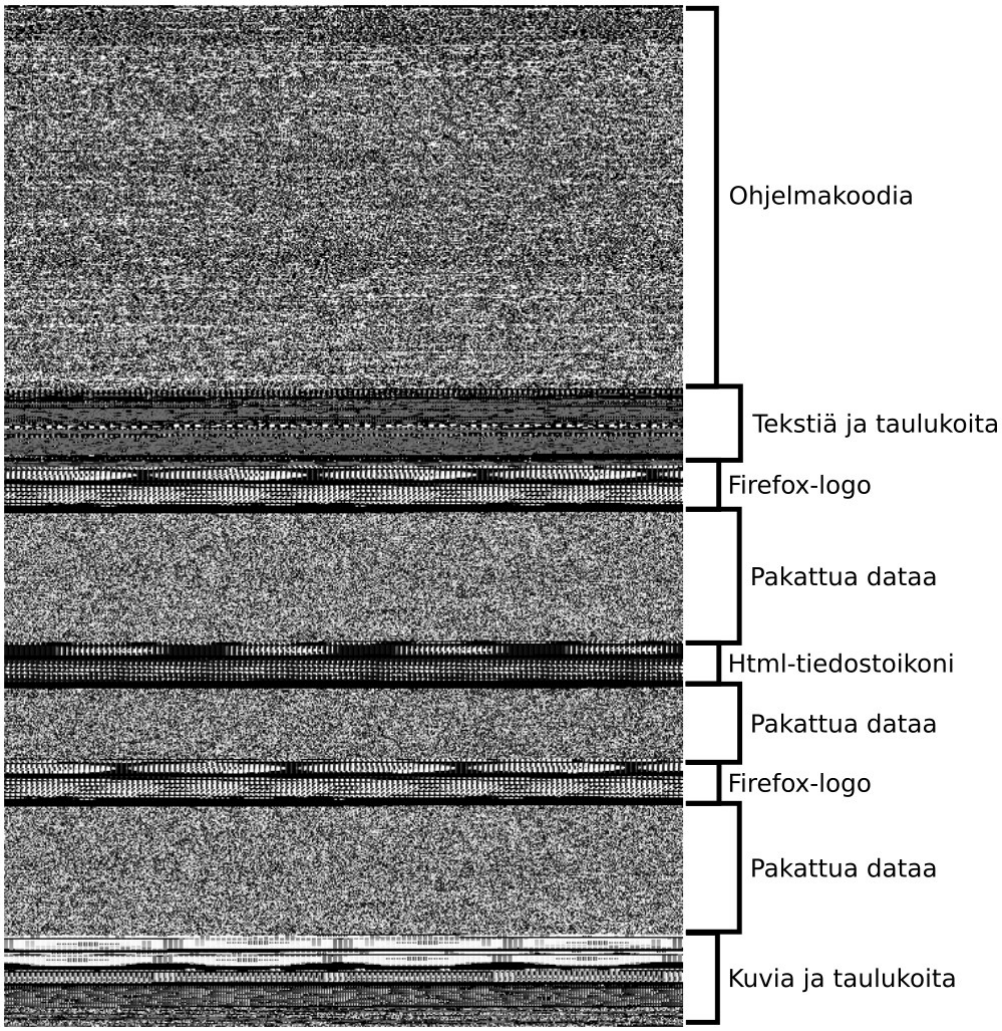
6) 

8) 

Kuva 15. *Digrafeja osista 4, 6 ja 8. Digrafit numeroitu osan mukaan.*

Kuvassa 15 olevat digrafit ovat keskitetty niin, että koordinaatiston origo on kuvan keskellä. Tämän avulla voidaan helpommin huomata, että arvo 0 vaikuttaa yleisimmeltä kuin muut arvot, jolloin digrafiin muodostuu kaksi viivaa x- ja y-akselin suuntaisesti. Kaikissa digrafeissa esiintyy myös kasaantumia origon oikealle puolelle. Nämä voivat olla pakkausalgoritmin jättämiä artefakteja [6].

Visualisointialgoritmien avulla päädyttiin kuvan 16 mukaiseen lopputulokseen, jossa osat ovat nimetty mahdollisten käyttötarkoituksiensa mukaan. Lopputuloksesta voidaan suoraan huomata tärkeitä rakenteellisia seikkoja, jotka voivat auttaa huomattavasti myöhemmin suoritettavassa takaisinmallinnuksessa.



Kuva 16. *Lopullinen muistikartta Firefox-ohjelmasta*

Saadun muistikartan avulla voitaisiin tehdä jatkotutkimusta eri osista, ja mahdollisesti takaisinmallintaa esimerkiksi pakattu data selkomuotoon. Eri osia voidaan myös verrata muihin löydettyihin muistikarttoihin ja näin löytää samankaltaisuuksia jopa eri tiedostotyyppien välillä, jos ne sattuvat käyttämään samankaltaisia tietorakenteita.

6. TULOKSET JA YHTEENVETO

Työssä perehdyttiin eri datavisualisoinnin algoritmeihin ja niitä käytettiin esimerkkitiedoston tutkimiseen. Esimerkkitiedosto pystyttiin jakamaan osiin eri visualisointien avulla ja osista pystyttiin erottamaan erilaisia tietorakenteita sekä päättelemään rakenteiden mahdollisia käyttötarkoituksia.

Digrafi sekä lineaarinen asettelu osoittautuivat hyödyllisiksi työkaluiksi niiden joustavuutensa ansiosta. Digrafilla pystyi erottamaan kaksi tietorakennetta toisistaan nopeasti ja tyypittämään tietorakenteita. Lineaarisen asettelun avulla pystyi tarkastelemaan suuriakin määriä dataa nopeasti ja erottelemaan sitä eri mielenkiinnon kohteisiin. Lineaarisesta asettelusta pystyi myös hahmottamaan kuvarakenteita ja säännöllisiä rakenteita suhteellisen tarkasti. Hilbert-kurvista oli tässä työssä rajatusti hyötyä ja se koettiin lineaarista asettelua epäselvemmäksi tarkastelussa, sillä se voi vääristää tietorakenteiden rajoja. Visualisoinnit antoivat tuntemattoman datan tarkasteluun hyvän lähtökohdan ja työstä voidaan nähdä kuinka yksinkertaisetkin visualisoinnit auttavat huomattavasti ohjelmistojen rakenteet tutkimisessa.

Jatkotutkimuksen kannalta olennaista on tutkia eri muuttujien vaikutusta visualisointeihin, sekä lisätutkimusta visualisoinnin eduista eri automaattisten työkalujen osana. Esimerkiksi digrafin askelväliä muuttamalla voi mahdollisesti paljastaa jotakin uutta. Abstraktien visualisointien tutkiminen voi myös koitua tehokkaaksi, kuten digrafin käytöllä on todettu. Visualisointien kanssa olisi voinut myös käyttää aikaisemmissa tutkimuksissa käytettyjä statistisia tarkistelutapoja, kuten khiin neliö -testiä, luomaan monipuolisempia visualisointeja [2]. Digrafia tulisi tutkia paremmin, sillä nykyinen tutkimus siitä on lähes olematonta.

LÄHTEET

[1] E. Eldad & E.J Chikofsky, Reversing : Secrets of Reverse Engineering, 2005 Saatavissa https://ebookcentral.proquest.com/lib/tampere/detail.action?docID=227440 (viitattu 24.3.2019)

[2] G. Conti et al., Automated mapping of large binary objects using primitive fragment type classification. Digital Investigation, 2010, Volume 7, S3–S12 p. Saatavissa: https://www.sciencedirect.com/science/article/pii/S1742287610000290 (viitattu 17.2.2019)

[3] C. E. Shannon, A Mathematical Theory of Communication, The Bell System Technical Journal. 1948, Volume 27, 379–423, 623–656 p.

[4] H. Wright et. al., Using visualization for visualization: An ecological interface design approach to inputting data. Computers & Graphics, 2013, Volume 37, 202–213 p. Saatavissa https://www-sciencedirect-com.libproxy.tuni.fi/science/article/pii/S0097849313000150 (viitattu 24.3.2019)

[5] A. Cortesi, Visualizing entropy in binary files. Saatavissa: https://corte.si/posts/visualisation/entropy/index.html (viitattu 17.4.2019)

[6] M. Pytel, Binary visualization explained. Saatavissa: https://codisec.com/binary-visualization-explained/ (viitattu 17.4.2019)

[7] Uncompressed RGB Video Subtypes, Saatavissa: https://docs.microsoft.com/fi-fi/windows/desktop/DirectShow/uncompressed-rgb-video-subtypes (viitattu 26.2.2019)

[8] A. Butz, Space Filling Curves and Mathematical Programming, Information and Control. 1968, Volume 12, 314–330 p. Saatavissa https://www.sciencedirect.com/science/article/pii/S0019995868903677 (viitattu 17.2.2019)

[9] ISO/IEC 8859-1:1998

[10] Mozilla Firefox. Saatavissa: https://www.mozilla.org (viitattu 19.4.2019)

[11] ISO/IEC 2382:2015. Saatavissa: https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en (viitattu 19.4.2019)